uniqname: _____

EECS 183 Winter 2019

Exam 2

Closed Book 3" x 5" notecard No Electronic Devices Closed Neighbor

All cell phones and other electronic devices, including smart watches, must be turned off and placed in your backpack

Instructions

1. After we indicate you may start, print your uniqname at the top of every page of this booklet.
2. After we indicate you may start the exam, verify that you have received every page of the exam. There are 19 pages in this booklet, including this cover page.
3. You may have and reference one 3" x 5" notecard.
4. Assume all code and code fragments are syntactically valid, unless otherwise specified.
5. Assume/use only standard C++11.
6. In all the given code, if any character looks like a space, it is one.
7. Write clearly. If we cannot read your writing, it will be marked wrong. This includes indentation and curly braces.
8. Sign below and print your name and UMID. 10 pts off if we cannot read your name or UMID.

"I have neither given nor received aid on this examination, nor have I concealed any violations of the Honor Code."

_____

SIGN your name above

_____

Clearly and legibly print your **full name (NOT uniqname)** above

_____

Clearly and legibly print your **UMID** above

uniqname: _____

Problem Number	Points Possible
Multiple Choice	100
Matching	20
Free Response 1	26
Free Response 2	26
Fill in the Blanks	28
	200

You can use the area below for working out problems.

uniqname: _____

Multiple Choice [5 points each]

Record your choices for all multiple choice questions using the circles next to each question. Fill the circle corresponding to your answer completely, like this: Ⓐ● Ⓒ Ⓓ Ⓔ

1. Which of the following statements about C++ arrays is **true**?	Ⓐ● Ⓒ Ⓓ

A. Arrays can be used to store elements with different data types.
B. Array elements are stored sequentially in memory.
C. You can determine the size of an array with the `.size()` function.
D. Arrays can be redeclared in order to change their size.

2. Consider the following function declaration: `bool compare(int &x, int y);` Which of the following code snippets contains valid function call(s)?	Ⓐ Ⓑ Ⓒ● Ⓔ

A.
```
int x = 10;
int y = 5;
cout << compare(x, y);
```

B.
```
int x = 1;
cout << compare(x, 1);
```

C.
```
cout << compare(2, 1);
```

D. Both A and B are valid function calls.

E. A, B and C all have valid function calls.

uniqname: _____

3. Which of the following statements about C++ classes is **false**?	Ⓐ Ⓑ Ⓒ Ⓓ

A. Classes can contain multiple variables with different data types.
B. One can define multiple class types with the same name.
C. All instances of a class will contain the same member variables and member functions.
D. One can limit direct access of data members in a class using the `private` keyword.

Consider the following code fragment, which will be **used in the next two questions**.

```
ifstream ins;
ins.open("data.txt");
int x = 0;
int count = 0;
while (ins >> x) {
  count++;
}
cout << count;
ins.close();
```

4. What prints if `data.txt` is opened successfully and contains: `1 2 3 # 4 5`	Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

A. 3
B. 4
C. 6
D. 10
E. 15

5. What prints if `data.txt` is opened successfully and contains: `1 2 3 4 5 6`	Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

A. 4
B. 6
C. 7
D. 15
E. 21

uniqname: _____

Consider the following code fragment:

```
string course = "EECS183";
for (int i = 3; i < course.length() - 2; i++) {
  course[i] = course[i+2];
}
cout << course;
```

6. What does the above code fragment print?	Ⓐ Ⓑ Ⓒ Ⓓ

A. `ECS183`
B. `EEC1833`
C. `EEC8383`
D. the result is unpredictable -- the above code makes an out of bounds array access

7. Which of the following statements about `C++` class constructors is **true**?	Ⓐ Ⓑ Ⓒ Ⓓ

A. Constructors must take on the same name as the class in which they are defined.
B. Only one constructor can be defined for each class.
C. The return type of a constructor is always `void`.
D. A and C are both true statements.

uniqname: _____

Consider the following class definition, which will be **used in the next two questions**.

```
class Hero {
  public:
    Hero();
    Hero(string n, char t, int p);
  private:
    string name;
    char type;
    int power;
};
```

8. Which of the following is/are valid declaration(s) for an instance of the `Hero` class?	A B C D E

A. `Hero spiderHam;`
B. `Hero spiderHam();`
C. `Hero("SpiderHam", 'D', 120);`
D. A and C are valid declarations.
E. A, B and C are valid declarations.

我觉得C少了 name，应该是这样的
Hero spiderHam("", , 120)

9. Assuming a `Hero` class instance called `spiderHam` has been declared, which of the following code statements would compile?	A B C D

A. `spiderHam.print();`
B. `cout << spiderHam.name << spiderHam.power << spiderHam.type;`
C. `spiderHam.write(cout);`
D. None of the above are syntactically correct.

uniqname: _____

Consider the following snippet of code:

```
int a[2][2] = {{0, 1}, {2, 3}};
cout << a[0][0] << a[1][1] << a[0][1] << a[1][0];
```

10. What does the above code print?	Ⓐ Ⓑ Ⓒ Ⓓ

A. 0123
B. 1032
C. 0312
D. 0213

11. Which of the following function declarations have valid parameters?	Ⓐ Ⓑ Ⓒ Ⓓ

A. `void array_function(int &arr[100]);`
B. `int array_function(int arr[100][100]);`
C. `string array_function(string arr[int size][int size]);`
D. B and C contain valid declarations.

12. True or False: the following snippet of code will correctly print all the values in `arr`:	Ⓐ Ⓑ

```
int arr[5] = {2, 4, 8, 16};
cout << arr << endl;
```

A. True
B. False

uniqname: _____

Consider the following C++ function declaration along with its RME:

```
/**
 * Requires: size <= declared size of data array
 * Modifies: nothing
 *  Effects: returns the number of elements in data > 0.
 **/
int count_positive(int data[], int size);
```

13. Which of the following function calls are valid test cases that may **expose bugs** in `count_positive`?	Ⓐ Ⓑ Ⓒ Ⓓ

A.
```
int data[3] = {-1, 1, 0};
cout << count_positive(data, 3);
```

B.
```
int data[] = {-3, -4};
cout << count_positive(data, 2);
```

C.
```
int data[3] = {-1, 1};
count_positive(data, 3);
```

D. A and B are valid test cases.

14. Which of the following statement(s) about C++ streams is/are **true**?	Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

A. One needs to #include the `fstream` library to use file streams.
B. One can declare multiple file input or output streams in the same program.
C. Only the standard input stream can go into a fail state.
D. A and B are both true statements.
E. A, B and C are all true statements.

uniqname: _____

Consider the following class definition, which will be **used in the next three questions**.

```
class Pair {
  public:
    int left;
    int right;

    /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: initializes left and right to zero.
      **/
    Pair();
    /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: initializes left and right to l and r, respectively.
      **/
    Pair(int l, int r);
};
```

15. Which of the following is a proper implementation of the non-default constructor as it should be written inside of Pair.cpp?	Ⓐ Ⓑ Ⓒ Ⓓ

A.
```
void ::Pair() {
    left = 0;
    right = 0;
}
```

B.
```
void ::Pair(int l, int r) {
    left = l;
    right = r;
}
```

C.
```
pair::Pair() {
    left = 0;
    right = 0;
}
```

D.
```
Pair::Pair(int l, int r) {
    left = l;
    right = r;
}
```

Consider the class definition of `Pair` above and the following program:

```
#include <iostream>
#include "Pair.h"
using namespace std;

void pair_func1(Pair &pair);

int main() {
  Pair p1(2, 6);
  pair_func1(p1);
  cout << p1.left << "," << p1.right << endl;
  return 0;
}

void pair_func1(Pair &pair) {
  int temp = pair.left;
  pair.left = pair.right;
  pair.right = temp;
}
```

16. Assuming that the non-default constructor for `Pair` has been implemented correctly, what does the above program print?	Ⓐ Ⓑ Ⓒ Ⓓ

A. `6,2`
B. `2,6`
C. `2,2`
D. `6,6`

Consider the class definition of `Pair` above and the following snippet of code:

```
Pair pairs[100];
```

17. What line of code prints the `left` value of the last pair in the array?	Ⓐ Ⓑ Ⓒ Ⓓ

A. `cout << pairs[100].left;`
B. `cout << pairs[99].left;`
C. `cout << pairs[left];`
D. `cout << pairs[left - 99];`

uniqname: _____

Consider the following snippet of code:

```
int x = 2;
for (int i = 0; i < 3; i++) {
  for (int j = i; j < 3; j++) {
    x += 1;
  }
}
cout << x << endl;
```

18. What value prints for x?	(A) (B) (C) (D) (E)

A. 4
B. 6
C. 8
D. 12
E. 14

19. True or False: An instance of a C++ class is always pass-by-reference when used as an argument in a function call.	(A) (B)

A. True
B. False

uniqname: _____

Suppose the definition of class `List` includes the following member function declaration:

```
/**
 * Requires: Nothing
 * Modifies: Nothing
 *  Effects: returns true if list is empty and false otherwise.
 **/
bool is_empty();
```

20. If `lst` is an instance of `List`, which of the following expressions checks if `lst` is empty?	Ⓐ Ⓑ Ⓒ Ⓓ

A. `is_empty(lst)`
B. `List.is_empty()`
C. `lst.is_empty()`
D. `List.is_empty(lst)`

lst.isempty

uniqname: _____

Matching [20 points]

Match the term on the left to the phrase on the right that best describes it. Write the letter on the empty line next to the phrase.

A. constructor

B. class

C. instance

D. encapsulation

E. fail state

F. scope resolution operator

G. interface

H. streams

I. public

J. private

I G ___ class members accessible outside of class scope

D ___ combining data and all functionality together in a class

F ___ identifies a function implementation as part of a specific class

G X ___ set of member functions that defines how to interact with instances

A ___ member function used to initialize data members

E ___ should be checked after stream extractions to ensure expected execution

J ___ prevents direct access of data members outside of a class

D ___ C++ keyword used to define custom or user-defined data types

C ___ object in memory that contains the properties defined by a specific class

H ___ used to manipulate input/output in a C++ program

The following class definition in the `Hero.h` file is relevant to the remaining questions on the exam.

```
// pre-defined hero types
const char DEFENDER = 'D';
const char ATTACKER = 'A';
const char HEALER = 'H';

class Hero {
  public:
     Hero() {
        name = "";
        type = DEFENDER;
        power = 0;
     }

     /**
      * Requires: t is one of DEFENDER, ATTACKER, HEALER
      * Modifies: name, type, power
      *  Effects: sets name, type and power to values in n, t and p, respectively
      **/
     Hero(string n, char t, int p) {
        name = n;
        type = t;
        power = p;
     }

     /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: returns the value of type.
      **/
     char get_type();

     /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: returns the value of name.
      **/
     string get_name();

     /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: returns the value of power.
      **/
     int get_power();

  private:
     string name;
     char type;
     int power;
};
```

The following class definition in the `Squad.h` file is relevant to the remaining questions on the exam.

```
#include "Hero.h"
const int MAX_SQUAD_SIZE = 10;

class Squad {
  public:
     Squad() {
        size = 0;
     }

     /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: returns the value of size.
      **/
     int get_size();

     /**
      * Requires: nothing
      * Modifies: heroes, size
      *  Effects: inserts a hero into the heroes array at the next empty
      *      position if current size is less than MAX_SQUAD_SIZE. At most two
      *      healers may exist in the squad. Therefore, if inserting h into the
      *      heroes array would cause it to contain more than two healers,
      *      it should not be inserted.
      *      Successfully inserting a hero should increase the size of the squad.
      *      Return true if hero is inserted and false otherwise.
      **/
     bool insert_hero(Hero h);

     /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: returns the number of HEALERs in the heroes array.
      **/
     int count_healers();

     /**
      * Requires: nothing
      * Modifies: nothing
      *  Effects: returns the sum of the power of all the heroes.
      **/
     int get_total_power();

  private:
     Hero heroes[MAX_SQUAD_SIZE];
     int size;
};
```

uniqname: _____

Free Response 1 [26 points]

Fill in the boxes and write the definition of the **`count_healers`** member function from the `Squad` class in the box below, following the RME from the class definition.

Note: assume your implementation is inside of `Squad.cpp`, which #includes `Squad.h`.

```
/**
 * Requires: nothing
 * Modifies: nothing
 *  Effects: returns the current number of HEALERs in heroes array.
 **/
```

int Squad::count_healers() {

```
for(int i=0; i< SIZE; i++){
  int number=0;
  if(heroes[i].get_type()=="HEALERS"){
    number++;
  }

  return number;
}
```

}

uniqname: _____

Free Response 2 [26 points]

Fill in the boxes and write the definition of the **`insert_hero`** member function from the `Squad` class in the box below, following the RME from the class definition.

Note: assume your implementation is inside of `Squad.cpp`, which #includes `Squad.h`.

```
/**
  * Requires: nothing
  * Modifies: heroes, size
  *  Effects: inserts a hero into the heroes array at the next empty
  *      position if current size is less than MAX_SQUAD_SIZE. At most two
  *      healers may exist in the squad. Therefore, if inserting h into the
  *      heroes array would cause it to contain more than two healers,
  *      it should not be inserted.
  *      Successfully inserting a hero should increase the size of the squad.
  *      Return true if hero is inserted and false otherwise.
  *   NOTE: your implementation should make use of the count_healers
  *      Squad member function.
  **/
```

[ ] [ ] ([ ]) {

[ ]

}

Fill In The Blank [28 points]

Fill in the blanks in the following program. Differences in whitespace (newlines and spaces) will not affect grading.

```
#include <iostream>
#include <fstream>
#include <string>
#include "Hero.h"
#include "Squad.h"
using namespace std;

/**
 * Requires: nothing
 * Modifies: file
 *  Effects: reads from file the values for name, type and power
 *       for each hero in a squad.
 **/

___________ read_squad(_______________ &file) {

    Squad team7;
    string name;
    char type;
    int power;

    _____________ (_________________ >> name >> type >> power) {

       ___________________.insert_hero(_____________(name, type, power));
    }
    return team7;
}
```

IMPORTANT: this problem continues on the next page.

uniqname: _____

```
/**
 * Sample run (bold text is user input):
 *   Enter name of file to read: heroes.txt
 *   Loading squad from heroes.txt file
 *   Your squad has 3 heroes with total power = 280
 **/

int main() {

  string filename;

  cout << "Enter name of file to read: ";
  cin >> filename;
  cout << "Loading squad from " << filename << " file" << endl;

  ifstream ins;

  ins.open(_____________________);

  _____________ team7 = read_squad(____________________);

  _____________.close();

  cout << "Your squad has " << _____________ • _____________ << " heroes "


       << "with total power = " << _____________ • _____________ << endl;


    return 0;

}
```